Title
Toward Exascale Earthquake Ground Motion Simulations for Near-Fault Engineering Analysis

Journal
Computing in Science & Engineering, 19(5)

ISSN
1521-9615

Authors
Johansen, Hans
Rodgers, Arthur
Petersson, N Anders
et al.

Publication Date
2017

DOI
10.1109/mcse.2017.3421558

Peer reviewed

Draft manuscript for Computing in Science and Engineering
May 11, 2017, revised June 6, 2017

Toward Exascale Earthquake Ground Motion Simulations for Near-Fault Engineering Analysis

Authors: Hans Johansen (LBNL), Arthur Rodgers (LLNL), N. Anders Petersson (LLNL), David McCallen (LBNL), Bjorn Sjogreen (LLNL), Mamun Miah (LBNL)

Abstract

We are modernizing the application SW4 for massively parallel time-domain simulations of earthquake ground motions in three-dimensional earth models to increase resolution and provide ground motion estimates for critical infrastructure risk evaluations. This development is greatly increasing the frequency content of computed ground motions including frequencies up to 10 Hz that are relevant to analysis of engineered structures (buildings, energy systems, bridges, dams, etc.). Simulations of ground motions from large (magnitude, M $\geq$ 7.0) earthquakes require domains on the order of 100-500 km and spatial granularity on the order of 1-5 m resulting in 100's of billions of grid points. Scaling proven algorithms from 10's of billions of grid points to 100's billions presents challenges with the domain decomposition and communication between nodes that must be overcome to take full advantage of next generation resources, on the path to exascale simulations. Surface-focused structured mesh refinement (SMR) allows for more constant grid point per wavelength scaling in typical earth models, where wavespeeds increase with depth. In fact, MR allows for simulations to double the frequency content relative to a fixed grid calculation on a given resource. We report improvements to the SW4 algorithm developed while porting the code to the Cori Phase 2 (Intel Xeon Phi) systems at the National Energy Research Scientific Computing Center (NERSC) at Lawrence Berkeley National Laboratory. Investigations of the performance of the innermost loop of the calculations found that reorganizing the order of operations can improve performance for massive problems. We demonstrate this capability with M 7 scenario earthquake simulations on the Hayward Fault and building response calculations in the eastern San Francisco Bay Area.

1. Introduction and Science Motivation

Earthquake ground motions pose an ever-present risk to engineered structures and the infrastructure that modern life depends on. Civilization has evolved in close proximity to active earthquake faults and sedimentary basins that amplify seismic motions. However, many cities of high ground motion risk have not experienced damaging motions due to long time intervals between large earthquake events. Ground motions are especially strong in the near-source region (< 10 km) of large (magnitude, M $\geq$ 7.0) events where peak ground accelerations and velocities greater than 1 g and 1 m/s are possible. Examples from the 1992 M 7.3 Landers,

California, 1999 M 7.6 Izmit, Turkey, 2003 M7.9 Denali Alaska and 2008 M 7.6 Chi Chi Taiwan earthquakes show such large near-fault motions. Figure 1a shows the ground acceleration, velocity and displacement of the fault-normal horizontal component of the 1992 Landers, California M 7.3 earthquake (Chen, 1995). In this near-fault region ground motions are heavily dependent on the specific nature of faulting, directivity, the location of asperities, the path that the seismic waves traverse (such as low seismic wavespeed sedimentary basins) and the shallow site geotechnical conditions (particularly vs30, the averaged shear-wave in the upper 30 m).

Furthermore, the response of buildings, bridges and other engineered structures in this near-fault region depends heavily on the nature of ground motions. This particular example of near-fault motions shows a strong one-sided velocity pulse and step in displacement that produces strong forcing to building foundations to the point of yielding components of the building.

Traditional empirically based earthquake hazard estimates depend on Ground Motion Predictions Equations (GMPE's), which are regression models derived from world-wide empirical data that estimate ground motion intensity as a function of magnitude, distance, faulting, path and site parameters. While GMPE's provide homogenized, smooth predictions of intensity and its variation, in some cases they do not capture the observed variability seen in data in the near-field. In Figure 1a the GMPE peak ground acceleration and velocity predictions for this record from two models are indicated (ASK14, Abrahamson et al., 2014; BSSA14, Boore et al., 2014). There are very few measurements of large earthquakes at close distances, hindering modeling efforts that try to capture the true nature and variability of near-fault motions. Figure 1b shows the magnitude (6.5-8.0) and distance (0-8 km) combinations of available strong-motion records from earthquakes from the Pacific Earthquake Engineering Research (Ancheta et al., 2014). As a result, we have precious few observations of large earthquakes within the critical near-fault region and this limits what can be done to evaluate how structures will respond to future earthquakes.

In the absence of empirical data for large earthquakes in the near-fault region, seismologists are using high-performance computing to simulate ground motions (Olsen et al, 2008; Aagaard et al., 2008, 2010; Ciu et al., 2013). In order to represent path and site effects caused by heterogeneity related to geologic structure (e.g. spatial geology variations, basin amplification, focusing, reverberations), methods must accurately describe three-dimensional wave propagation. To obtain ground motions at frequencies comparable to observations (static to 5-10 Hz), models require discretization as small as 2-10 m over domains that include the entire rupture length of M 7-8 earthquakes, (100-500 km). For example, a 3D seismic simulation spanning the rupture of an M 7 earthquake requires 10's to 100's of billions of grid points (10^{10} - 10^{12}) and numerical wave propagation simulations on this scale require massively parallel computations. For a fixed domain, doubling the frequency requires halving the grid spacing in three dimensions. This requires $2^3 = 8$ more grid points relative to the reference case and twice as many time steps to simulate the motion. Consequently, the computational effort increases

by a factor of 16 to compute the same motion while doubling the frequency content. This represents an extremely challenging barrier to simulating high-frequency earthquake ground motions.

In this study we describe "SW4" (*Seismic Wave 4th-order*), a summation-by-parts finite difference code for parallel simulations of seismic wave propagation. We describe how SW4 is being enhanced to run simulations on the next generation of HPC systems. We show how SW4's implementation of mesh refinement provides a remarkable reduction in the computational effort required to simulate a given earthquake. This is followed by a description of preliminary efforts to optimize SW4 for the Cori Phase 2 architecture at the National Energy Research Scientific Computing Center (NERSC) at Lawrence Berkeley National Laboratory. We conclude by showing an example of how these improvements can be applied to scenario earthquake simulations of the Hayward Fault and building response in the eastern San Francisco Bay Area.

2. Scientific Features of SW4

It is important to consider the performance and optimization of a science application like SW4 in the context of its science simulation goals. For our Cori Phase 2 target platform, there were three science considerations that affected our choice: mesh refinement, a regional scale 2 Hz simulation, and coupling ground motion to building engineering simulations. These are important, as they provided the science context and problem sizing for the node-level optimization goals.

2.1 Mesh Refinement

Cities tend to be built in sedimentary basins where the ground is flat and where water and transportation corridors (including rivers and harbors) are accessible. These areas are generally characterized by low seismic shear wavespeeds (typically 200-500 m/s) corresponding to poorly consolidated soils. Shear wavespeeds increase with depth to values of 3000 m/s at 5 km depths and to 3500 m/s at 25 km (the deepest extents of most crustal earthquakes). Consequently, shear wavespeeds increase by a factor of 7 or more across the depth extent of computational domains. This fact confounds seismic modeling because numerical methods require a certain minimum number of grid points per wavelength of the shortest wave, corresponding to the maximum frequency shear-waves, while the time step depends on the Courant–Friedrichs–Lewy (CFL) condition which is driven by the fastest wavespeed, corresponding to compressional waves in the uppermost mantle (typically ~8000 m/s).

To overcome this challenge, SW4 has a mesh refinement capability (Figure 2) where the user specifies depths at which the grid spacing doubles relative to the overlying region. This allows the simulation to maintain a roughly stable number of grid points per shortest wavelength throughout the computational domain and increase the time-step relative to the finer grid.

Both these factors make the calculation far more efficient than a standard uniform mesh.

Note that finite and spectral element methods rely on a similar strategy of increasing element size as seismic wavespeeds increase with depth, however in these methods the user must build and verify the unstructured mesh. SW4 generates the mesh automatically based on the input surface topography and user-specified mesh refinement depths. (Petersson and Sjogreen, 2016).

2.2 Regional-Scale Earthquake Ground Motion Simulations

To assess the initial scaling of the SW4 code to a science goal for Cori Phase 2, we performed regional-scale simulations of an M 7.0 earthquake on the Hayward Fault in the San Francisco Bay Area. The Hayward Fault is capable of earthquakes of this size, including the last known rupture (estimated M 6.8) on October 21, 1868. Strong shaking was experienced throughout the East Bay (Boatwright and Bundock, 2008) but recording instruments were not yet developed. The current seismic hazard assessment for Northern California identifies the Hayward Fault as the most likely fault to rupture with a M 6.7 or greater event before 2044 at 14% (Field et al., 2015). We performed ground motion simulations with both a simplified plane-layered (one-dimensional, 1D) and more complex three-dimensional (3D) model representing geologic structure from the United States Geological Survey (USGS, 2017). Previous studies have used this model for modeling moderate recorded events (Rodgers et al., 2008; Kim et al., 2010), large past earthquakes such as the M 7.9 1906 San Francisco (Aagaard et al., 2008) and possible Hayward Fault scenarios (Aagaard et al., 2010). Previous 3D simulations of Hayward Fault scenarios generally modeled frequencies below 0.5-1.0 Hz. In this study we modeled frequencies from static to 2.5 Hz.

Figure 3 shows snapshots of the magnitude of the ground velocity at 10 seconds and the peak ground velocity (both in ShakeMap color palette) for the 1DREF (left) and 3DUSGS (right) models. For this vertical strike-slip event the ground motions are symmetric across the fault for the 1DREF model as expected (Figure 3ab). However, the 3DUSGS model shows more complex response with sedimentary basins delaying, trapping and amplifying ground motions. For example, the eastern side of San Pablo Bay, the Dublin-Pleasanton-Livermore Tri-Valley and Delta have deep sediments that amplify motions. Mount Diablo has high wavespeed material and lower amplitudes. Importantly, we see dramatic differences across the Hayward Fault with larger motions on the eastern side (Figure 3cd). The East Bay Hills (EBH) east of the Hayward Fault shows higher peak motions than areas west of the fault at the same distance from the fault. These differences have been seen in previous simulations, including those in Aagaard et al. (2010) and arise from lower wavespeeds in the upper crust east of the Hayward Fault. Further investigations are needed to verify and evaluate these differences and improve the 3D model with waveform-based optimization.

2.3 Coupling Geophysics (hazard) to Engineering (risk)

A principal goal of our application development efforts is to, for the first time, develop computationally based ground motion estimates with sufficient frequency resolution to inform engineering risk evaluations for a spectrum of critical infrastructure. The approach we are following is a direct coupling of ground motion simulations to nonlinear finite element model infrastructure simulations including the effects of structural yielding and damage. The overall approach is illustrated in Figure 4 whereby the ground motions across the surface of the SW4 domain are used as point-wise forcing functions for the evaluation of the response of infrastructure systems. For each infrastructure evaluation, a representative canonical infrastructure model is analyzed for the input ground motion, and peak response variables are saved to provide an intensity map of risk.

As an example, for the SW4 regional scale Hayward Fault simulations described above, the risk model for a representative 40 story steel moment frame building, analyzed using the Nevada nonlinear finite element program (McCallen and Larsen, 2003) is shown in Figure 5. The structural risk is characterized in terms of peak interstory drift in the building, i.e. the peak relative displacement between two adjacent floors. Such risk maps provide new insights into the regional scale variation of risk for major earthquake events, and can be used to investigate the relationship between geophysical parameters (fault rupture parameters, rupture directivity, geologic structure, site response, etc.) and infrastructure response.

3. Optimization for NERSC Cori Phase 2

3.1 Algorithm and Computational Kernels

SW4 consists of several double-precision computational kernels (loops) to simulate high-fidelity seismic waves in realistic geological settings (Petersson and Sjogreen, 2016):

- A *forcing function,* which simulates the earthquake source in space and time.
- A *stress calculation* ("*RHS*")*,* which applies compact finite difference stencils to displacements and material properties to calculate the divergence of the stress tensor. Variants include Cartesian or *curvilinear* metrics for terrain, with and without stretching for super-grid far-field treatment.
- Applying *super-grid damping* near far-field domain boundaries.
- Enforcing the *boundary conditions* on the free surface and far-field boundaries
- A *predictor-corrector* procedure for updating displacements in time.
- Additional kernels for mesh refinement boundary matching conditions, attenuation of the seismic waves, and I/O, which are critical for the science simulations and workflow, were not examined for this study.

Domain decomposition is horizontal (see Figure 2); each MPI process gets a full vertical column of a fixed size, from surface to the bottom of the domain (a "*pencil*"). Because SW4 is a finite

difference code, it requires "ghost points" to be communicated between MPI processes before the next *RHS* calculation can be performed. For this purpose, each MPI task exchanges information with its neighbors by calling MPI_SendRecv().

3.2 NERSC Cori Phase 2 Architecture

NERSC's new Cori system has 9,688 Intel Xeon Phi (code name Knights Landing, "KNL") nodes in its Phase 2 deployment, with a theoretical peak of 29.1 PFlops. Three of the important features of the Xeon Phi processor are (1) its 16 GB of MCDRAM high-bandwidth memory ("HBM") and shared 1MB L2 cache per 2 cores, (2) dual 512-bit vector processing units (VPU's) per core, and (3) its large number of cores and hardware threads per chip (up to total 272 threads), but with lower clock speed (1.4 GHz) than the Cori Phase 1 Intel Xeon processors (2.3 GHz).

For the SW4 application developers, the Xeon Phi architecture implications were studied through a roofline model (Jeffers et al., 2016), which estimates peak FLOPS based on the arithmetic intensity (FLOPS/byte) of an application. To obtain a significant fraction of throughput performance, developers must (1) make effective use of MCDRAM (with 4x DDR bandwidth) and L2 cache as a "stream" of application data to cores, (2) transform loops into 8-double-wide SIMD AVX512 instructions, and (3) tile and thread loops to utilize all available cores and threads, while minimizing cache misses and conflicts that result in additional data motion.

3.3 Porting Approach

The starting point for SW4 optimization was the full science-ready application, with MPI-only domain decomposition (no threading), and most loops implemented in FORTRAN with a unit-stride component (*c,i,j,k*) data layout for the 3 spatial components c=1,2,3 of the vector displacement variable, which previously had given 30 percent of peak performance on Intel Xeon architectures.

Through the NERSC Exascale Science Application Program (NESAP 2016), the SW4 team joined the community of people preparing science application codes for large-scale runs on Cori Phase 2. This program proved critical to providing access to DOE and Cray/Intel experts, performance tools, early access to Intel Xeon Phi nodes, and a venue to share information and ask for help. The steps we took to prepare for porting included:

Best Practice	SW4 Team Approach
1. Extract performance-critical kernels.	Created *SW4Lite* test driver with variants of *RHS* loops, with OpenMP pragmas.
2. Generate test inputs/outputs that are	Identified a 128^2 x 2000 (33M grid point)

representative of the largest target science problem.	problem that could fit on a single Xeon Phi node 16GB MCDRAM.
3. Create correctness tests for this problem with an error tolerance.	Comparison scripts on time series output to detect errors greater than roundoff.
4. Automate/accelerate, as much as possible, the code-build-test-measure process and tool chain.	Git/CMake tools; command line options for kernel variants; job scripts with MPI/OpenMP and affinity settings; timers around most expensive code sections.
5. Prioritize optimizations based on effort vs. potential performance gains.	Prioritized on-node optimizations: simple threading, array reordering, loop tiling. Deferred complex threading, loop fusion.
6. Assess performance bottlenecks against benchmarks and past improvements.	Used roofline model and Intel toolchain to profile and identify sub-optimal code.

The SW4 team targeted a scaled-down science problem, which would be equivalent of a 5 Hz simulation distributed across all of Cori's nodes, but run as a single node, 33M grid point problem, with no internode communication. This seemed justified, based on previous weak scaling studies on up to 131,072 MPI-tasks and 8192 nodes of the IBM BGQ system "vulcan" at LLNL.

3.4 Performance Analysis and Modernization Process

We decided to create a stand-alone test driver, called "SW4lite" with the 4 variants of the RHS loops that existed in the full SW4 application. This allowed us to try intrusive performance changes without rewriting large amounts of code, and to evaluate changes quickly for benefit vs. effort. We prioritized on-node optimizations, which included: porting FORTRAN loops to C; implementing simple threading using OpenMP pragmas; reordering data structures from *(c,i,k,j)* to *(i,j,k,c)*, to improve vectorization data alignment; and an initial pass at loop tiling.

We introduced changes in a way that allowed us to maintain before/after comparisons. For example, benchmark input files and scripts with run-time configurations were version controlled in a git repository; additional build-specific information such as compiler flags and library versions were also documented with any significant performance results. This allowed us to compare performance gains to previous results, and regression-test when performance degradation was seen with new versions of compilers or libraries. This was particularly important as the Cori system moved from the initial "white box" system to early access, and finally pre-production, during which system software and libraries changed frequently.

A few simple, specific techniques were used to improve the rate of code changes in SW4lite. For example, FORTRAN loops were converted to C, so that the array data could easily be reordered from *(c,i,k,j)* to *(i,j,k,c)* at run-time, as was the data type, to help with Intel-specific types and alignment declarations. Converting to C-based loops allowed us to quickly change ordering, array offsets, and data types, without extensive code changes. Reordering was key for changing the SW4 "array of structures" ordering to a "structure of arrays". The reordering resulted in a factor of 3 speedup for the calculation of the super-grid damping term ("SG" bars in Figure 6) because component 'c' of the damping term can be calculated from the same component of the displacement, thus reducing a stride length of 3 to unity. The reordering also turned out to be beneficial for the stress calculations ("Step" bars in Figure 6), even though each component of the stress involves all three components of the displacement, which Intel's compilers can more effectively vectorize, given that the "component" unit stride length of 3 is not a multiple of vector length (8) and thus requires gather/scatter operations. Spreading the components across *(i,j,k)*-indexed memory also allows for more efficient streaming from MCDRAM memory (NERSC 2016, Jeffers 2016). Command line arguments exposed which variant of the RHS kernel to run, including reverse-order indexing, with curvilinear or Cartesian mapping, etc.; this allowed incremental, faster builds to be used for small code changes, and reduce compile times (which were significant, more than 30 minutes for the most complex kernels – see below). For things like loop tiling and changing OpenMP pragmas (such as "parallel for" and "simd" clauses), recompiling was unavoidable, so we attempted to maintain multiple kernel versions and expose them as command-line options whenever possible.

To assess opportunities for further performance improvements, we followed the steps advised by the NERSC Exascale Science Access Program (NESAP 2016), which were designed to assess the specific performance-limiting features of the application code (see NERSC 2016, and other KNL optimization techniques). For example, using "numactl" we were able to see if performance improved using MCDRAM vs. main DDR memory. Similarly, running AVX512, AVX2, and scalar-only experiments helped understand how much the code was benefiting from vectorization. Experiments varying MPI ranks with OpenMP and hardware threads explored how much idle time or latency was affecting the code (see Figure 6).

4. Results and Lessons Learned

4.1 Optimization Results

Over the course of our efforts, we were able to speed-up the most important SW4lite kernels on Xeon Phi by a factor of ~2x. Although this would have been achievable in FORTRAN, the code variations and performance gains were easier in C++. There were a number of steps that needed to be taken to achieve this; the easiest by far was adding OpenMP pragmas to thread the *k*-index loops, which allowed for more cores to be used in a given MPI rank. Next, declaring AVX512 aligned data types and forcing vectorization of loops had two effects: aligned data is

more effectively streamed into L1 cache, and vectorization achieves its promised 8x speed-up. This also required the *(i,j,k,c)* data ordering, which was only implemented in C++ through SW4's class for multi-dimensional arrays (see NERSC 2016). Finally, loop tiling in the *i*- and *j*-index directions improved cache locality and reuse. Effort was spent optimizing both Cartesian and curvilinear loop kernels with super-grid stretching and boundary conditions, with similar benefits. Code example 1 shows examples of the kernels in FORTRAN and C++.

In addition to the above optimizations, a substantial amount of performance improvement could still be gained with more complex transformations. During the profiling process, we discovered that the Intel compiler was exiting its vectorization pass prematurely. Run-time performance improved significantly by overriding the default compiler optimization limits, and waiting *45 minutes* while compiling the most FLOP-intensive numerical kernel. This was a unique compile-time bug in the Intel 2017 compiler, which is slated to be fixed in the Intel 2018 compiler. After waiting for the completed and optimized compilation, we discovered that the Xeon Phi vector registers were over-subscribed, and the loops could benefit from both cache blocking and fissioning (to reduce register pressure). In addition, the Intel Advisor tools showed that the separate predictor-corrector updates had relatively low *arithmetic intensity* (Williams et al., 2009), and could benefit from fusing with the RHS loops (which could benefit from loop fission, due to excessive register pressure). However, this would be an invasive operation involving temporary arrays with the potential for thread safety complications. As a result we have postponed such experiments to a later phase of the project.

4.2 Lessons Learned

Our primary learnings center around the architecture-specific optimization process, where a deeper understanding of the Xeon Phi many-core, vectorized, and memory-limited architecture is required. Our process and tool chain were very effective in creating an environment in which performance optimizations could be explored, while remaining relevant to the actual science application goals at full scale. The Intel 17 tool chain (Advisor and VTune, specifically) provided many insights and relatively easy data collection. The Intel Advisor Roofline feature was very helpful in setting expectations and priorities for code improvements in critical parts of the code, as well as sifting out parts of the code that wouldn't matter at scale. We were hampered by the difficulty with compilation times; fast turn-around between code changes and performance analysis is a critical step in the application optimization process.

Opportunities for improvements in the code optimization process are numerous; in hindsight, we could have taken some additional steps before delving in:

- Identify a relevant code benchmark, such as *miniFD* or *HPGMG* (Williams et al., 2016), and establish their performance for a similar-size problem. Often fine-tuned techniques have already been established in benchmark codes for OpenMP threading, vectorization, cache blocking, etc.

- Refactor your code to approximate the benchmark, by commenting out loop or other application-specific constructs, and verifying performance results like what the benchmark obtains.
- Refactor back to the correct application code, taking small steps to evaluate correctness with each addition, noting performance changes and profiling the application to understand its efficiency.

Had we followed this process, we would have quickly found that the kernel performance didn't meet our expectations based on stencil benchmark codes, and we could have more quickly identified problematic issues with compilation and vectorization, and made progress sooner.

5. Discussion and Next Steps

In this study, we report performance improvements of SW4 that enable more efficient simulations, and are paving the way towards *exascale* simulations of earthquake ground motions and application to seismic engineering risk assessment. The critical lessons learned include: (1) establishing a smaller problem that is representative of the performance of a large scale science goal, but easily run on just a few nodes; (2) creating a team and tool chain for evaluating optimizations, such as OpenMP, data layout, and loop tiling, across the large performance parameter space; and (3) utilize both high-level and detailed performance profiling tools that are specific to the target architecture. On this last point, we benefited extensively from the resources provided the NERSC Exascale Science Access Program, which provided both expertise and Intel Xeon Phi profiling tools and training. The greatest benefit has come from the SW4 team working in conjunction with optimization experts and a development team that spans expertise in performance optimizations, algorithms, and science goals; this also represents the greatest risk as limited resources are pulled in different directions within the project. In an ideal world, code exemplars and expertise would be available for specific patterns, such as finite difference stencil applications, and code templates or domain-specific tools would provide a solid starting point for architecture-optimal code.

Future work to improve application performance and scaling will focus on porting SW4lite kernels back into SW4, and incorporating OpenMP, data layout, and vectorization improvements. Additional performance improvements might be realized through loop fission and improved threading, but additional evaluations are needed. Single precision performance will surely boost performance, since AVX512 can demonstrably obtain 2x the performance, but only if the loss of accuracy is acceptable for the seismic risk assessment application. Finally, we want to explore communication-hiding approaches to better overlap MPI with OpenMP threading, and improve the performance of the mesh refinement algorithm.

As computational capabilities advance, the ability for performing simulation based hazard and risk estimates that accurately reflect the underlying physics and site specificity of earthquake

phenomenon will provide a new approach to ensuring infrastructure safety. As a result of the high computational demands, it will be essential to fully exploit emerging exascale computations as a foundation to transformational risk assessments.

Appendices

A.1 Data and Resources

Earthquake ground motion data in Figure 1 was obtained from the Pacific Earthquake Engineering Research (PEER) Center (Ancheta et al., 2014).

SW4 is an open-source seismic simulation code developed at Lawrence Livermore National Laboratory (Petersson and Sjogreen, 2016) and distributed by the Computational Infrastructure for Geodynamics, available at: https://geodynamics.org/cig/software/sw4/

The 3D seismic model of the San Francisco Bay Area was obtained from the United States Geologic Survey (USGS, 2017).

Figures were made with ObsPy (Krischer et al., 2015) and Generic Mapping Tools (Wessel et al., 2013).

A.2 Acknowledgements

We are grateful for support from the US Department of Energy Exascale Computing Project and access to the Cori Phase-II cluster at National Energy Research Scientific Computing Center (NERSC) at Lawrence Berkeley National Laboratory. We are also grateful for a Computing Grand Challenge allocation on Quartz at Lawrence Livermore National Laboratory. Arben Pitarka (LLNL) provided assistance with earthquake rupture models and simulations. This work was performed in part under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

A.3 References

Aagaard, B. T., T. M. Brocher, D. Dolenc, D. Dreger, R. W. Graves, S. Harmsen, S. Hartzell, S. Larsen, K. McCandless, S. Nilsson, N. A. Petersson, A. Rogers, B. Sjogreen, M. L. Zoback (2008). Ground-Motion Modeling of the 1906 San Francisco Earthquake, Part II: Ground-Motion Estimates for the 1906 Earthquake and Scenario Events, Bull. Seismo. Soc. Am. 98(2), 1012-1046, doi: 10.1785/0120060410.

Aagaard, B. T., R. W. Graves, A. Rodgers, T. M. Brocher, R. W. Simpson, D. Dreger, N. A. Petersson, S. C. Larsen, S. Ma, and R. C. Jachens (2010). Ground motion modeling of Hayward fault scenario earthquakes, Part II: Simulation of long-period and broadband ground motions, Bull. Seismo. Soc. Am., 100(6), 2945-2977, doi: 10.1785/0120090379.

Ancheta, T. D., R. B. Darragh, J. P. Stewart, E. Seyhan, W. J. Silva, B. S.-J. Chiou, K. E. Wooddell, R. W. Graves, A. R. Kottke, D. M. Boore, Tadahiro Kishida and Jennifer L. Donahue (2014). NGA-West2 Database, *Earthquake Spectra*, 30:3, 989-1005, doi: 10.1193/070913EQS197M

Abrahamson, N., W. Silva and R. Kamai (2014). Summary of the ASK14 Ground Motion Relation for Active Crustal Regions, *Earthquake Spectra*, 30:3, 1025-1055, doi: 10.1193/070913EQS198M

Boatwright, J. and H. Bundock (2008). *Modified Mercalli Intensity Maps for the 1868 Hayward Earthquake Plotted in ShakeMap Format*, Open-File Report 2008-1121, United States Geological Survey, http://www.usgs.gov/.

Boore, D. M., J. P. Stewart, E. Seyhan and G. M. Atkinson (2014). NGA-West2 Equations for Predicting PGA, PGV, and 5% Damped PSA for Shallow Crustal Earthquakes, *Earthquake Spectra*, 30:3, 1057-1085, doi: 10.1193/070113EQS184M

Chen, X. (1995). Near-Field Ground Motion from the Landers Earthquake, *California Institute of Technology, Earthquake Engineering Research Laboratory*, Report No. EERL 95-02, Pasadena, California, USA.

Cui, Y., E. Poyraz, K.B. Olsen, J. Zhou, K. Withers, S. Callaghan, J. Larkin, C. Guest, D. Choi, A. Chourasia, Z. Shi, S.M. Day, J.P. Maechling, and T.H. Jordan (2013). Physics-based seismic hazard analysis on petascale heterogeneous supercomputers, *Procs. Supercomputing Conference*, 2013.

Field, E. H. and 17 others (2015). Long-Term Time-Dependent Probabilities for the Third Uniform California Earthquake Rupture Forecast (UCERF3) *Bull. Seism. Soc. Am.*, 105, 511–543, doi: 10.1785/0120140093

Jeffers, J., J. Reinders, and A. Sodani (2016). Intel Xeon Phi Processor High Performance

Programming, 2nd Edition, Knights Landing Edition. Elsevier, ISBN: 9780128091944.

Kim, A., D. S. Dreger and S. Larsen (2010). Moderate Earthquake Ground-Motion Validation in the San Francisco Bay Area, Bull. Seismol. Soc. Am., 100, 819-825.

L. Krischer, T. Megies, R. Barsch, M. Beyreuther, T. Lecocq, C. Caudron, J. Wassermann (2015) ObsPy: a bridge for seismology into the scientific Python ecosystem, *Computational Science & Discovery*, 8(1), 014003, DOI: 10.1088/1749-4699/8/1/014003

McCallen, D.B., S. Larsen (2003), NEVADA – A simulation environment for regional scale estimation of ground motion and structural response, Lawrence Livermore National Laboratory Report UCRL-ID-152115.

NERSC - Getting Started and Optimization Strategy (2016). More information at: http://www.nersc.gov/users/computational-systems/cori/application-porting-and-performance/getting-started-and-optimization-strategy/

NESAP – the NERSC Exascale Science Application Program (2016). More information at: http://www.nersc.gov/users/computational-systems/cori/nesap/

Olsen, K.B., S. M. Day, J. B. Minster, Y. Cui, A Chouasia, R. Moore, P. Maechling, T. Jordan (2008), TeraShake2: simulation of Mw7.7 earthquakes on the southern San Andreas fault with spontaneous rupture description, *Bull. Seismol. Soc. Am.* 98, 1162-1185, doi: 10.1785/0120070148.

Petersson, N. A. and B. Sjogreen (2016). SW4 User Guide, version 1.18, LLNL-SM-662014.

Petersson, N.A. and B. Sjogreen (2015). Wave propagation in anisotropic elastic materials and curvilinear coordinates using a summation-by-parts finite difference method, J. Comput. Phys. 299, pp. 820-841, http://dx.doi.org/10.1016/j.jcp.2015.07.023.

Petersson, N.A. and B. Sjogreen (2014). Super-grid modeling of the elastic wave equation in semi-bounded domains, Comm. Comput. Phys. 16, pp. 913-955.

Petersson, N.A. and B. Sjogreen (2012). Stable and efficient modeling of anelastic attenuation in seismic wave propagation. Comm. Comput. Phys. 12(1), pp. 193-225, DOI: 10.4208.

Rodgers, A., N. A. Petersson, S. Nilsson, B. Sjogreen, K. McCandless (2008). Broadband waveform modeling of moderate earthquakes in the San Francisco Bay Area and preliminary assessment of the USGS 3D seismic velocity model, Bull. Seismol. Soc. Am., 98, 969-988, doi: 10.1785/0120060407.

Sjogreen, B. and N.A. Petersson (2012). A fourth order accurate finite difference method for the elastic wave equation in second order formulation, J. Sci. Comput. 52(1), pp. 17-48, DOI: 10.1007/s10915-011-9531-1.

United States Geologic Survey (2017). 3-D Geologic and Seismic Velocity Models of the San Francisco Bay Region, http://earthquake.usgs.gov/data/3dgeologic, last accessed May 11, 2017.

Wessel, P., W. H. F. Smith, R. Scharroo, J. F. Luis, and F. Wobbe (2013). Generic Mapping Tools: Improved version released, *EOS Trans. AGU*, 94, 409-410, DOI: 10.1002/2013EO450001

Williams, S., A. Waterman, and D. Patterson (2009). Roofline: an insightful visual performance model for multicore architectures. Communications of the ACM, 52(4):65–76, 2009.

Williams, S., et al. HPGMG-FV Git repo. http://bitbucket.org/hpgmg/hpgmg.

A.4 Figures and Captions

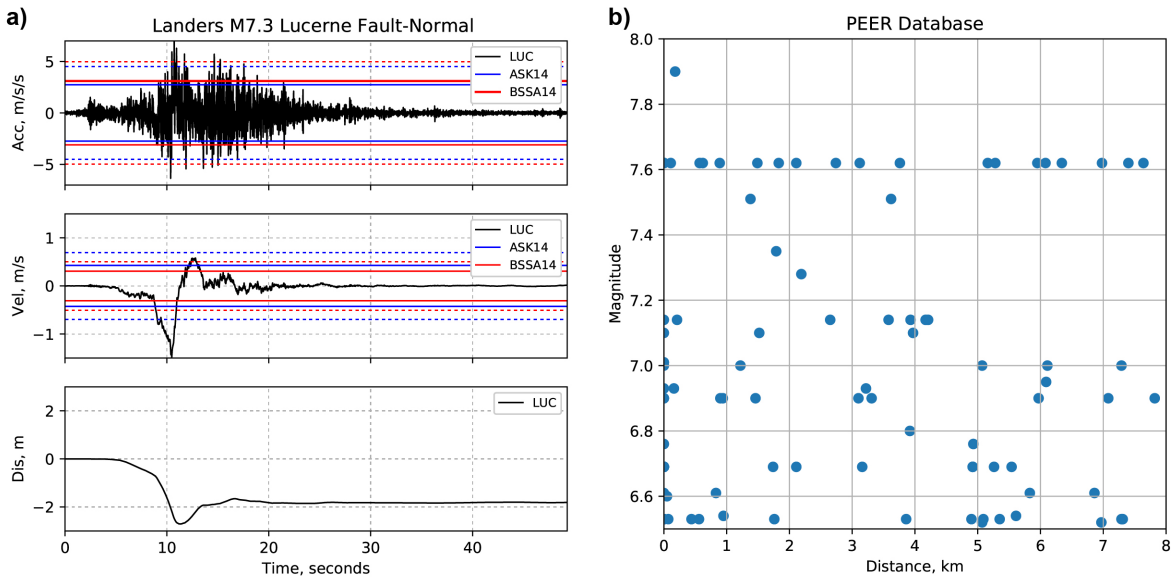

Figure 1. (a) Acceleration (top), velocity (middle) and displacement (bottom) for the 1992 Landers, California M 7.3 earthquake at station Lucerne 2.2 km from the fault. Also shown are Ground Motion Prediction Equation predictions for this near-fault record: solid colored lines shown the median prediction of peak ground acceleration (top) and velocity (middle) and dashed lines show standard deviation of the ASK14 and BSSA14 models (see text). **(b)** Magnitude-distance combinations of strong-motion records in the PEER NGA West-2 database for magnitudes 6.5-8.0 and distances 0-8 km.

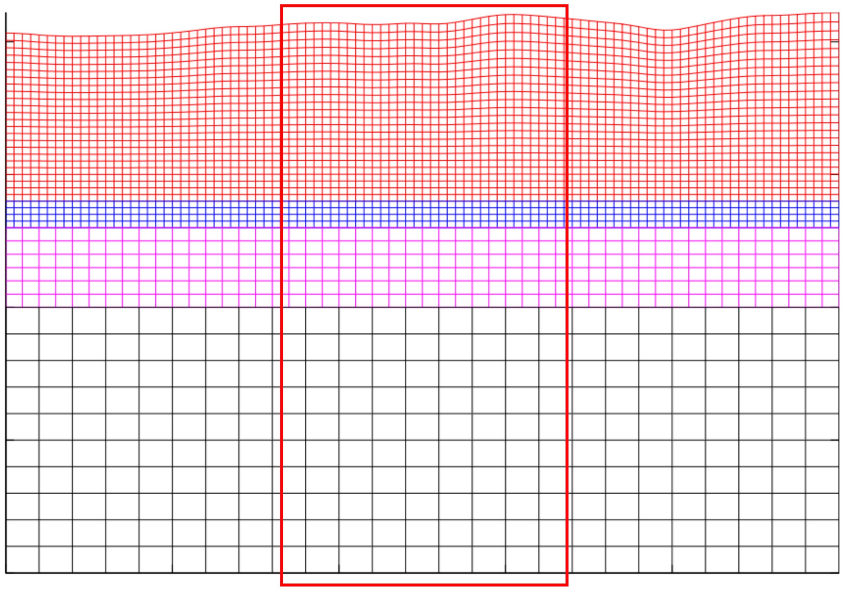

Figure 2. In SW4, each MPI-task is assigned a pencil-shaped computational sub-domain (red box). Each pencil contains a part of the total domain, starting at the free surface boundary on top, and extending to the far-field boundary on the bottom. The top grid (red) is the most computationally expensive due to both the large number of grid points, and the metric terms in the curvilinear mapping. Lower layer grids (blue, pink, & black) have increasingly fewer points and are Cartesian, which allows for a less complex finite difference stencil.

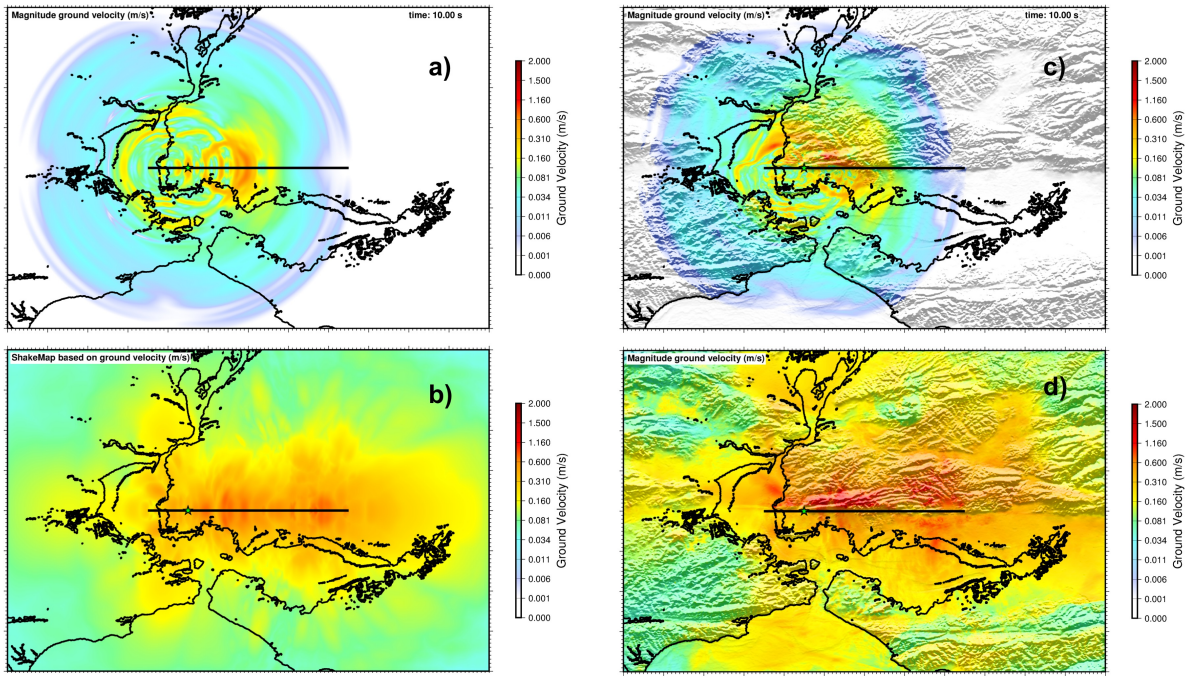

Figure 3. Simulated ground motions for an M 7.0 Hayward Fault earthquake showing: **a)** the magnitude of ground velocity at 10 s and **b)** ShakeMap based on peak ground velocity for the 1D model; and **c)** and **d)** the same for the 3D model. Geologic features are identified in d): Delta, Sacramento-San Joaquin Delta; EBH, East Bay Hills; MD, Mount Diablo; SPB, San Pablo Bay; Tri-V, Dublin-Pleasanton-Livermore Tri-Valley.

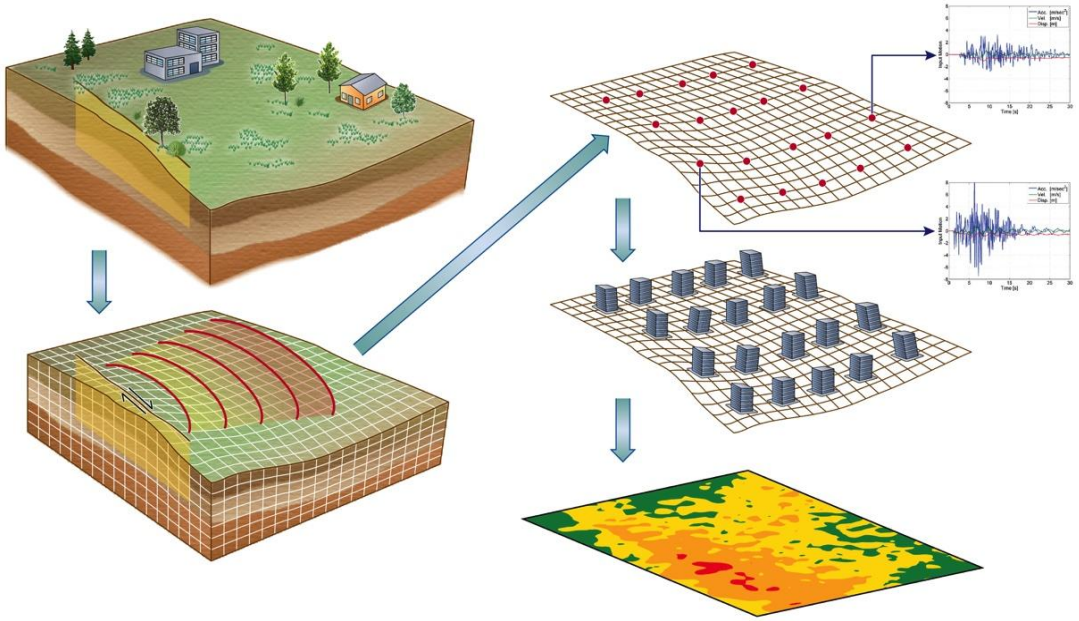

Figure 4. Transforming hazard into risk: utilizing ground motion estimates from a regional scale geophysics model to drive infrastructure assessments.

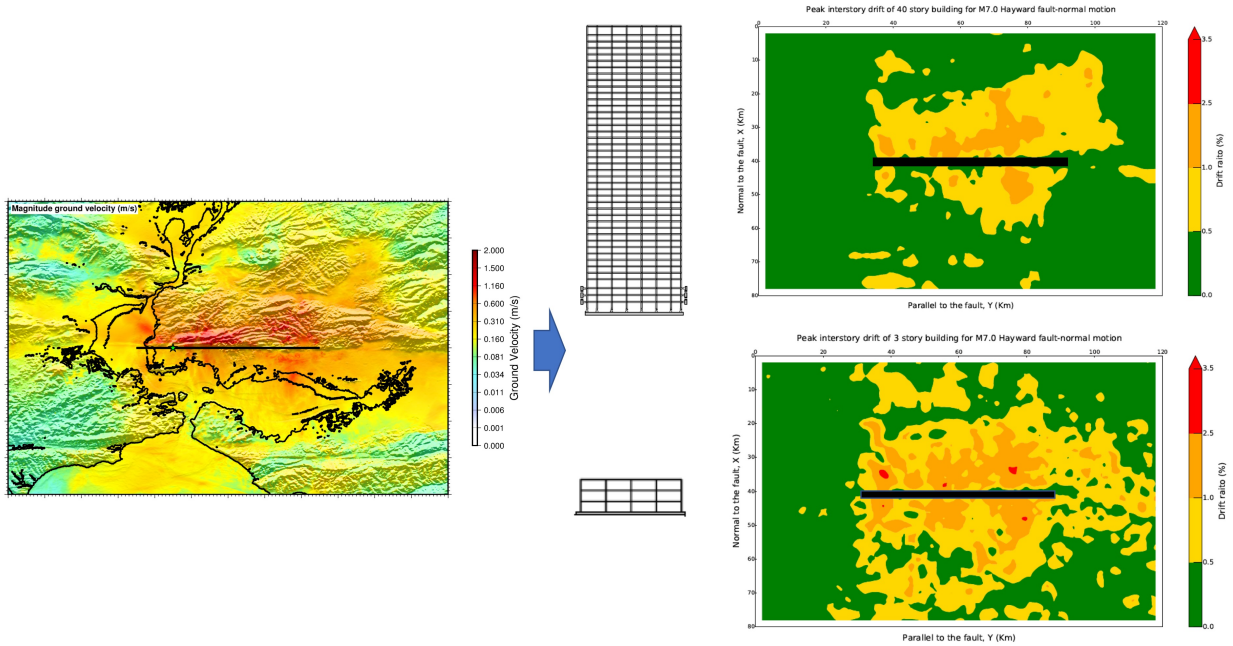

Figure 5. Utilizing ground motions to evaluate regional infrastructure risk; Hayward Fault rupture scenario resulting demands/risk on representative buildings (in terms of peak interstory drift).

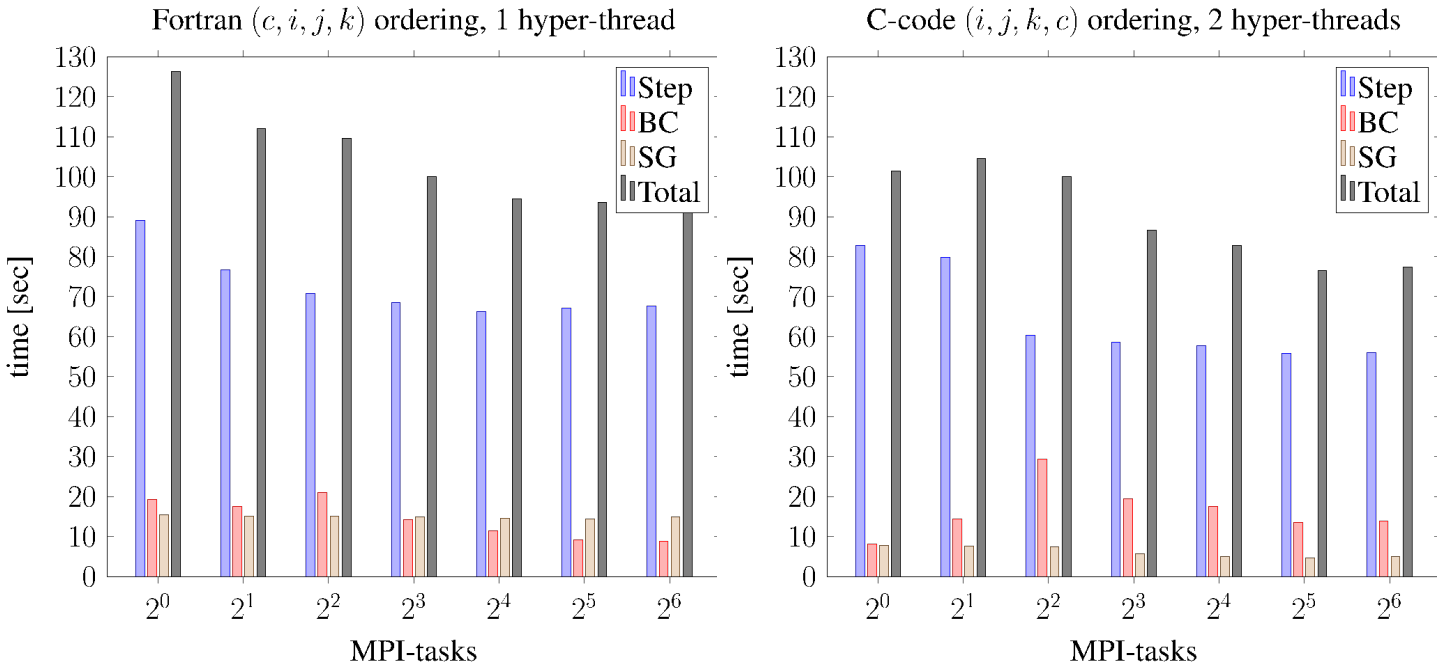

Figure 6. Initial performance studies (pre-optimization) of the different SW4lite kernels, looking at trade-offs between array data layouts, and MPI/OpenMP/hardware threads. The Fortran kernel with component-first indexing was generally slower; the C version with *(i,j,k,c)* ordering showed better results with two hardware threads, but performance varied more erratically.

FORTRAN	C++

real*8 u(3,ilo:ihi,jlo:jhi,klo:khi)
...
do k=klo,khi
  do j=jlo+2,jhi-2
  do i=ilo+2,ihi-2
...
r1 = r1 + i144*(la(i-2,j,k)
  *(u(2,i-2,j-2,k)- u(2,i-2,j+2,k)
  + 8*(-u(2,i-2,j-1,k)+u(2,i-2,j+1,k)))
  - 8*la(i-1,j,k)
  *(u(2,i-1,j-2,k)-u(2,i-1,j+2,k)
  - 8*(u(2,i-1,j-1,k)-u(2,i-1,j+1,k))))
...
	float_sw4* __restrict__ a_u;
	#define u(c,i,j,k) \
	  a_u[base3+i+ni*(j)+nij*(k)+nijk*(c)]
	...
	#pragma omp for
	for(k=k1; k<=k2; k++)
	  ... // Create tiles over i,j chunks
	  for(j=jfirst+2; j<=jlast-2; j++)
	#pragma simd
	#pragma ivdep
	  for(i=ifirst+2;i <= ilast-2; i++)
	...
	r1 = r1 + i144*(la(i-2,j,k)
	  *(u(2,i-2,j-2,k)-u(2,i-2,j+2,k)
	  + 8*(-u(2,i-2,j-1,k)+u(2,i-2,j+1,k)))
	  - 8*(la(i-1,j,k)
	  *(u(2,i-1,j-2,k)-u(2,i-1,j+2,k)
	  - 8*(u(2,i-1,j-1,k)-u(2,i-1,j+1,k))))
	...

Code example 1. Example of FORTRAN (left) code used in *SW4Lite*, and reimplementation in C++ (right). The differences include reordering array indices using a macro and adding OpenMP parallel and vectorization pragmas. Intel's C++ compiler is able to achieve similar performance to FORTRAN, and can make certain loop transformations easier, like cache blocking with tiling.

Keywords

Computational Seismology
Structural Mechanics
Earthquake Simulations
Exascale Computing

Author Bios / emails

Dr. Hans Johansen is a computer systems researcher in the Applied Numerical Algorithms Group at Lawrence Berkeley National Laboratory
hjohansen@lbl.gov

Dr. Anders Petersson is an applied mathematician in the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory
petersson1@llnl.gov

Dr. Arthur Rodgers is a seismologist in the Atmospheric, Earth and Energy Division at Lawrence Livermore National Laboratory
rodgers7@llnl.gov

Dr. David McCallen is a research affiliate at Lawrence Berkeley National Laboratory and an Associate Vice President in the University of California Office of the President
david.mccallen@ucop.edu

Dr. Bjorn Sjogreen is an applied mathematician in the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory
sjogreen2@llnl.gov

Dr. Mamun Miah is a postdoctoral fellow in Earth and Environmental Sciences at Lawrence Berkeley National Laboratory
mmiah@lbl.gov